RESPONSE TO OFFICE ACTION

Applicant very much appreciates that the rejections raised in the first office action have been withdrawn in view of the amendments and the arguments set forth in the response to the office action. Claims 6, 8, 10, 11, 12 and 15 are rejected as being anticipated by the Bradshaw article cited in the office action.

Summary of the Bradshaw System

Bradshaw is indeed directed to transactional system, and more particular to a system in which transactions can span multiple databases. The problem so be solved is the occurence of concurrent global transactions which may lead to non-serializable global histories (Abstract, end of first paragraph).

For the sake of reference, we quote the Wikipedia entry on Serializability:

> Serializability is the major criterion for the correctness of concurrent transactions' executions (i.e., transactions that have overlapping execution time intervals, and possibly access same shared resources), and a major goal for concurrency control. As such it is supported in all general purpose database systems. The rationale behind it is the following: If each transaction is correct by itself, then any serial execution of these transactions is correct.

An algorithm to maintain global serializability is proposed. The multidatabase system has the purpose of giving the illusion that the client application is accessing a single database system (1 Introduction, first paragraph).

For this purpose, the databases are organised as a hierarchy, as shown e.g. In Fig. 2 and the associated explanation in the text: the multidatabase transaction manager (MTM2) controls the local transaction

managers (LTM) via associated multidatabase processing agents (MTA).

A number of assumptions is made in order for the system to work as desired ("Assumptions" at the end of section 3. "Composite multidatabase model"). The last assumption, at the end of section 3, is very important in the current context:

> Finally, no two subtransactions at any multidatabase server or component database system are part of the same global transaction. This assumption guarantees that the runtime composition lattice for any global transaction always takes the form of a strict hierarchy.

In other words, only one subtransaction per process is allowed for the same global transaction.

Features of the present invention that are not shown in the Bradshaw System

For the same global transaction, there may be more than one subtransaction per process

As shown in the above summary, Bradshaw allows only one subtransaction per process (section 3, last paragraph). The present invention explicitly allows multiple subtransactions per process, as reflected e.g. in claim 6 by the wording "a second invocation of the at least one respective service by a transaction resulting in the creation of a second transaction local to...".

Subtransactions are defined differently

Bradshaw's "subtransactions" are not really subtransactions: section 1, paragraph 6: "The multidb server at Agency A treats the forwarded subtransaction like any other global transaction". Also see Bradshaw's section 6, point 3 'Global Aborts': "Each of the algorithms guarantees... through the pre-emption of non-serializable transactions by global aborts". This implies that subtransactions are not really subtransactions as in the present invention: in the invention the root (parent) rollback will lead to subtransaction rollback although the inverse is not true: if a subtransaction fails then the parent can still try alternative subtransactions.

It is impossible to implement the present system on the base of the Bradshaw system

Bradshaw claims that their system was built with Encina, a classical Transaction Processing (TP) monitor (section 6, second paragraph). The present invention is designed to surpass the limitations of products like Encina; the present invention cannot be realized with or on top of Encina!

The invention realizes global atomicity rather than global serializability

Bradshaw describes a system for guaranteeing global serializability, wheras the invention implements global atomicity - something different! The invention does not need, want nor claim global serializability. The major strength of the invention is precisely that it breaks global serializability (by using compensation/undo).

35 U.S.C. 102(a) rejections

Claim 6

Please recall, from the summary of the Bradshaw paper, that only one subtransaction per process is allowed for the same global transaction.

However, claim 6 states that both the first and the second invocation are invocations of the same service, implemented by the same process. This is clear from the claim language, since the claim states (paragraphs 2, 3 and 4):

- each process ... implementing at least one respective service ...
- a first invocation of the at least one respective service ... the first local transaction being a child of the invoking transaction ...
- a second invocation of the at least one respective service ... the the first local transaction being a child of the invoking transaction ... local transaction being a child of the invoking transaction ...

From this it follows that

- both invocations refer to the same service (since the wording is "the ... service" and not e.g. "a ... service").
- both the first and second local transaction are a child of the same invoking transaction.

Therefore, such a pair of invocations cannot exist in Bradshaw at all, since the Bradshaw assumptions explicitly require that, for one global transaction only one invocation or subtransaction per process is allowed.

Therefore, Bradshaw does not disclose this feature, and even teaches against it.

For this reason, the situation described in the next claim feature (paragraph 5)

- each process characterized in that if the first transaction and the second transaction conflict but are both children of a same invoking transaction, then the first transaction and the second transaction are not executed concurrently

can never arise in the Bradshaw system: The Bradshaw system, as stated above, does not allow the existence of two conflicting transactions, invoked in the same service, that are children of the same invoking transaction.

The section of Bradshaw cited by the examiner with relation to this feature (section 5.1, lines 4-9) does not apply: this section, beginning with

> Recall that a component database system C is said to have a rigorous history if whenever ...

is simply a citation of the general definition of the term "rigorous history". This definition mentions two transactions, but does not place any condition on these transactions.

The invention, however, introduces the additional condition that the conflicting transactions are the children of the same invoking transaction and allows a situation which is explicitly forbidden according

to Bradshaw.

Therefore, Bradshaw does not disclose this feature, and even teaches against it.

Summarizing the above, the essential features of claim 6 are not disclosed by Bradshaw and, as follows from the constraints placed on the Bradshaw system by the assumptions, can never come into existence. **In consequence, the subject matter of claim 6 is neither disclosed nor suggested by Bradshaw.**

Neither does Gupta disclose the abovecited feature of paragraph 5 of claim 6, as explained in the response to the first office action, and **therefore the combination of Bradshaw and Gupta does not suggest this feature either.**

Claim 8

The rejection of claim 8 cites different sections of Bradshaw and reads them on the different features of the claim. However, these different sections relate to different systems, on the one hand to the Bradshaw system itself, and on the other hand to related work from other authors and other database systems. In the present case, the Bradshaw system is designed to overcome the shortcoming of the related art (the necessity for maintaining common or central ticket data, see below). Therefore, someone skilled in the art would not incorporate the related art features described as disadvantageous into the Bradshaw system which is designed to eliminate exactly those feature!

Claim 8 recites a number of features, among which the following (paragraphs 5 and 6):

- (5) propagating from a first process to a second process a message indicative of a globalCommit operationwith respect to a root transaction, said message also indicative of a number or identifying list of invocations which the first process has made to the second process on behalf of the root transaction;
- (6) within the second process, comparing the number or list indicated in the message with a count or list within the second process of the number or list of invocations which have been

made on behalf of the root transaction;

The examiner cites section 2: Related work, column 3, as being relevant for these features. Closer inspection shows, however, that this section shows exactly the opposite of the claim features: It describes the working of concurrency control in multidatabase systems. Concurrency control is accomplished by forcing component databases to update a common ticket data item (col 3, lines 9-14). This requires the existence of a centralized global serialization graph (lines 17-20).
The invention, in contrast, explicitly states, in the preceding feature, that each process works independently of any centralized component (emphasis added):

(4) each process making scheduling and recovery decisions independent of any centralized component triggered by invocation of a service of another process, each process further characterized in that each transaction local thereto is independently handled at the process, each process making scheduling and recovery decisions independent of any centralized component, the method comprising the steps of:

Therefore, the section 2: Related work part of Bradshaw, whicht describes the use of a common ticked and central global serialization graph, contradicts what is claimed.

Furthermore, the applicant is unable to locate the features B and C in the cited section and invites the examiner to point out where exactly the subject matter of the paragraphs 5 and 6 cited above is disclosed!

Summarizing the above, the essential features of claim 8 are not disclosed by Bradshaw. **In consequence, the subject matter of claim 8 is neither disclosed nor suggested by Bradshaw.**

Neither does Gupta disclose the abovecited feature of paragraphs (5) and (6), as explained in the response to the first office action, and **therefore the combination of Bradshaw and Gupta does not suggest this feature either.**

Claim 10

Claim 10 is directed to a method corresponding to the system of claim 8. The same arguments set forth regarding paragraphs (4), (5) and (6) of claim 8 apply to claim 10 as well.

The examiner also cites Bradshaws section 5.2: Composite Timestamp Ordering Algorithm, as being relevant. This, at closer inspection, is however not the case: The second paragraph of this section (beginning with "The CTO algorithm assigns ...") shows the underlying elements required by the Composite Timestamp Ordering Algorithm, i.e.

- Transactions are assigned timestamps. Timestamps are inherited by subtransactions.
- Transactions are tagged as multicellular, or untagged. This tag is a boolean property stating whether a transaction is delegated from another multidatabase or not. This "tagged" property is inherited by subtransactions. Consequently, the algorithm on the following page uses the "untagged" boolean property of a transaction (line 4).

Thus, all that is passed or inherited from one transaction to another is one timestamp, and one boolean value.

In contrast, the invention claims that what is propagated is

> a message indicative of a globalCommit operation with respect to a root transaction, said message also indicative of a number or identifying list of invocations which the first process has made to the second process on behalf of the root transaction;

which is something completely different.

The information propagated according to the invention also serves a different purpose. Whereas the information used by Bradshaw is used together with a global concurrency scheduler of the cell, the information propagated according to the invention servers to eliminate such a global entity.

In more detail, Bradshaw mentions timestamps (used as serialization orders) that are propagated and

compared at commit time. The invention however propagates invocation counts instead, with an entirely different purpose: the purpose is to detect 'missed invocations', not serialization orders among those invocations. The inventive information propagation does not allow nor intend to detect any relative orders among subtransactions, but rather wants to assert that all subtransactions were taken into account at commit time.

In consequence, the subject matter of claim 10 is neither disclosed nor suggested by Bradshaw.

Since Gupta does not disclose key features of claim 10 (analogous to claim 8) either, **the combination of Bradshaw and Gupta does not suggest this feature either.**

Claim 11

Claim 11 differs from claim 10 in the last paragraph, covering the complementary alternatives resulting from the list comparison. The same arguments as for claim 10 apply here as well.

Claim 12 and claim 15

The examiner considers the timestamp propagated in the Bradshaw system as being the same as the "information about the actual work being committed".

The timestamp, however, is a chunk of information that is generated by the transaction system itself, for example, when a transaction request is generated or processed. Timestamp information is used by the CTO algorithm for ensuring serializability.

According to the invention, the information conveyed during globalCommit is not used to decide any transaction outcome. Instead, this information is used by external agents like a human administrator. In Bradshaw, the mentioned information during globalCommit is the serialization order among component processes, used by the transaction system to decide on rollback or commit. This is a crucial difference.

In more detail, as explained in the response to the first office action, according to the invention, information about the actual work being committed is distributed, in order to aid in resolving failures once they have occurred. This information is only passed along and stored by the system, as long as everything functions normally. Only in the cause of a failure is this information used, at a higher level, e.g. e.g. by a human, in order to "reorder the parts of the broken puzzle". This is made more explicit in system claims 13 and 14, and corresponding method claims 16 and 17.

For example, the system level information that is, according to the state of the art, returned by a failed transaction, may be something like "there has been a system failure with bank X". This is not helpful at all.

Thus, according to the invention, the information about the actual work such as "payment of customer xyz with credit card No. 123 at bank X has not been completed, although the order has been confirmed..." is also available and can be used to clean up the remaining inconsistencies, which would be much more difficult without this information. During ordinary operation of the transaction system, this information is not used, and the transaction system will work just the same if this information is missing.

Bradshaw does not consider the problem of recovery after a failure by passing around and using information about the actual work being committed. Thus, claims 12 and 15 are neither disclosed nor suggested by Bradshaw.

35 U.S.C. 103(a) rejections

In view of the above arguments establishing novelty of the independent claims, no further argumentation is required with regard to dependent claims 7, 9, 13, 14, 16, and 17.

Claim 18

The logic of the office action cannot be followed for the following reasons:

According to paragraph 3, the rejection based on Gupta was withdrawn as necessitated by the amendment.

The amendment of claim 18 introduced the following features into claim 18:

> wherein the root invocation (transaction) propagates the concurrency preferences with each or any child invocation it makes; and
>
> wherein the propagated concurrency preferences at any level in the root invocation's invocation hierarchy specify the extent to which shared resource access is desired or allowed or denied among descendant invocations of the root invocation or user and other, concurrent invocations who are also descendants of the same root.

Therefore, the withdrawal of the rejection due to the introduction of these features implies that these features are not disclosed in Gupta.

However, the presently pending office action states, on page 12, that exactly these features are disclosed by Gupta. This contradicts the withdrawal.

The applicant respectfully asks that this point of view be reconsidered. For this reason, the arguments showing that the abovecited features are not disclosed by Gupta are restated:

These feature state that the concurrency preferences are for descendants of the SAME root, which constitutes a difference with Gupta: Whereas Gupta's isolation levels take care of invocations among different transactions, according to the invention conflicts are addressed that are caused within the same overall (root) transaction.

Gupta does not consider the problem of conflicts occurring between different descendants of the same root transaction, and fails to provide the inventive solution. Thus, the abovecited features of claim 18 are neither disclosed nor suggested by Gupta.

As explained in the introduction, **Bradshaw** does not allow, for the same global transaction, more than one subtransaction per process. Therefore the problem solved by the invention

- does not even arise in the Bradshaw system, and
- would not be solved by any feature of Gupta.

And in consequence, claim 18 is not anticipated by the combination of Bradshaw and Gupta.

Claim 23 and dependent claims

Please recall the fundamental structure of the Gupta system: It provides for a central, transactional server for connecting client applications that are not transaction-aware (for more details and references, please refer to sections 3 and 3.1 of the previous response to office action). The clients, since they do not implement transactional behavior, do not provide for a globalCommit or globalAbort functionality. When a transaction (maintained by the central server) is aborted, then the necessary reversal of client actions is achieved by the server invoking compensating functions in the client. The compensating functions are generated and stored on the server on behalf of the client.

Please recall further that an invocation within a transaction will carry some token that identifies the transaction of the calling service. This is the transaction context. So, a service that receives an invocation from another service can detect any existing transaction by the presence of this token.

Combining Bradshaw and Gupta does not lead to the features of claim 23

In the Gupta system, the transaction context is maintained by the central transactional server. The Gupta clients or applications, being unaware of transactions, do not know about transaction contexts and are not programmed to process them.

Therefore, the following features of claim 23 cannot be disclosed by Gupta:

> An invocation by a remote client also containing partial or complete information indicating or

containing said client's context or contexts;

The reason is that the Gupta client (application) does not have a transaction context. It cannot create an invocation that contains the transaction context. Please note that the term "context" cannot be interpreted in an arbitrarily broad manner, since the preceding claim feature states that the contexts of the remote clients are transactions contexts.

Furthermore, reading the Gupta system on the abovecited feature constitutes a direct contradiction of the preceding claim feature

Some or all such remote clients having one or more associated contexts or transaction contexts

This features states that the remote clients have transaction contexts (that are also called simply "contexts"), whereas the Gupta clients, as explicitly stated, are not transaction aware and thus do not have a context.

In consequence, the combination of Bradshaw and Gupta does not lead to the invention according to claim 23.

The teachings of Bradshaw and Gupta are contradictory and incompatible

Furthermore, this combination would not have been contemplated by one of ordinary skill in the art, since the purposes of the two systems are incompatible, and the combination of features of the two systems is senseless. The reasons for this are:

The whole point of Gupta is making the service handling the applications transaction-unaware so the applications do not need a transaction context of standardized undo hooks. Thus, "globalAbort" and/or "globalCommit" messages are not needed and not present when servicing the applications or within the applications.

Therefore, adding a transaction context (as taken from Bradshaw) leads to unnecessary overhead, and

there is, on purpose, no use for it. It also defeats the purpose and benefit of Gupta's invention entirely. Indeed, Gupta is ideal for what it is designed for: centralized coordination of transaction-unaware applications. No globalCommit/globalAbort messages are sent to services.

Combining Bradshaw and Gupta the other way around does not make sense either: if you have a transaction context (as in Bradshaw and in the present invention) then you can rollback or undo based on this context: you just say "globalAbort transaction123". Moreover, this is the ONLY information you need, whereas Gupta needs the names of undo operations in a centralized place. Gupta also needs the service-specific (i.e. application-specific) parameters for each undo operation. This adds even more global knowledge.

So the present invention allows only localized knowledge, and particularly does NOT require the actual names and parameters of the undo operations to be returned or collected in a centralized place (because the name is always, by the protocol, "globalAbort" with the transaction identifier as argument). So the invention allow local scheduling, whereas Gupta does not.

Moreover, Gupta's "undo-from-centralized-scheduler" requires that the scheduler not only know the name of undo operations, but also their relative orders! This drastically increases complexity at the global level and can't possibly be called "localized knowledge" any more. The present invention assumes that this is known at the service level – thus localized only.

Another way in which the invention allows local knowledge is by allowing the service to undo on its own (after timeout). Gupta cannot do this because it has no transaction protocol between the service and the centralized scheduler. Consequently, timeouts at the service level would cause problems with possibly intermediate and contrary decisions at the centralized scheduler level.

In summary, Bradshaw already provides its own recovery mechanism (in the form of the transaction-oriented "globalAbort" handling),

- 1. Adding another recovery mechanism (Gupta's) that purposely eliminates the transaction mechanism for the clients defeats the purpose of the Bradshaw system, and so no one skilled in

the art would be motivated to do so.

- 2. Even IF this combination were made, this does not lead to the invention (where the clients are transaction-aware, contrary to Gupta).

Respectfully submitted,

/s/

Carl Oppedahl
PTO Reg. No. 32746